

Lecture 2: Java Basics

김 강 희

khkim@ssu.ac.kr

목 차

- 이론: 자바 문법 기초
 - ❖ “Object” 클래스
 - ❖ “Exception” 클래스
- 실습:
 - ❖ Matrix 클래스에 finalize 메소드 적용하기
 - ❖ Matrix 클래스에 Exception 클래스 적용하기
 - ❖ Matrix 클래스의 메소드들 이해하기

“Object” 클래스

- 자바에는 “Object” 라는 이름의 클래스가 존재 함
 - ❖ `java.lang.Object` 패키지에 정의되어 있음
- 모든 클래스의 수퍼클래스임
 - ❖ Object 클래스를 상속한다는 별도의 명세는 불필요함
- 다음 메소드들을 정의하고 있음
 - ❖ `finalize()` : 해당 클래스의 객체가 쓰레기로서 수집되기 직전에 해당 클래스의 `finalize`가 호출됨. 만약 서브클래스가 `finalize`를 재정의(`override`)하면 수퍼클래스의 `finalize`가 호출됨.
 - ❖ `clone()` : 생성자가 호출 비용이 큰 경우에 생성자 대신에 사용할 수 있음(단, `Cloneable interface`가 정의된 경우에 한해). `clone()`보다는 복사 생성자를 사용할 것을 권장함
 - ❖ `wait()`, `notify()`, `notifyAll()` : 쓰레드 동기화에 사용됨
 - ❖ `toString()` : 객체의 문자열 표현을 생성함
 - ❖ `hashCode()`, `equals()` : 쌍으로 사용되는 메소드들로서 두 객체가 동일한 객체인지 확인함. (예: `x.equals(y)`)

실습: Main1()

```

3  ▶ public class Main {
4  ▶     public static void main(String[] args) throws Exception { // why "throws Exception"???
5      |         main1(args);
6      |     }
7
8      public static void main1(String[] args) throws Exception { // why "throws Exception"???
9      |     Matrix m = new Matrix( cy: 10, cx: 10);
10     |     for (int i=0; i<999; i++)
11     |     |     m = new Matrix( cy: 10, cx: 10);
12     |     System.gc();
13     |     System.out.println("nAlloc=" + m.get_nAlloc());
14     |     System.out.println("nFree=" + m.get_nFree());
15     | }

```

nAlloc=1000
nFree=868

```

3  ▶ public class Matrix {
4  ▶     private static int nAlloc = 0;
5  ▶     private static int nFree = 0;
6  ▶     protected void finalize() throws Throwable { // why 'protected'?
7      |         super.finalize();
8      |         nFree++; // count the number of freed objects
9      |     }
10     public int get_nAlloc() { return nAlloc; }
11     public int get_nFree() { return nFree; }

```

nAlloc=1000
nFree=999

nAlloc=1000
nFree=567

Exception 처리

- Exception 처리의 필요성

- ❖ 자바 코드 상에서 에러가 발생한 즉시 사용자에게 에러 발생 지점과 에러 원인 메시지를 출력하여 디버깅을 도와주기 위함

- 동작 방식

- ❖ 자바 언어는 메소드들의 예외(또는 에러) 조건의 처리를 도와주기 위해 exception이라는 메커니즘(또는 클래스)를 제공함
- ❖ 한 메소드에서 에러가 발생하면 그 지점에서 적절한 exception 객체를 생성하여 throw함
 - 해당 메소드는 메소드 이름 뒤에 “throws ...Exception”이라는 문구를 반드시 포함해야 함 → 메소드 signature에 포함됨
 - ...Exception 이름은 그 메소드 안에서 발생 가능한 모든 exception들을 cover할 수 있는 general exception class 이어야 함
- ❖ Throw된 exception 객체는 발생 지점 이후에 처음으로 만나게 되는 try-catch 블록에서 처리됨
 - Catch 블록에서는 exception 객체의 타입을 식별하여 원하는 exception 객체만 catch할 수 있음
 - Throw된 exception 객체를 포함하는 superclass의 exception 타입을 명시하면 보다 많은 exception들을 하나의 catch 블록에서 처리할 수 있음

“Exception” 클래스

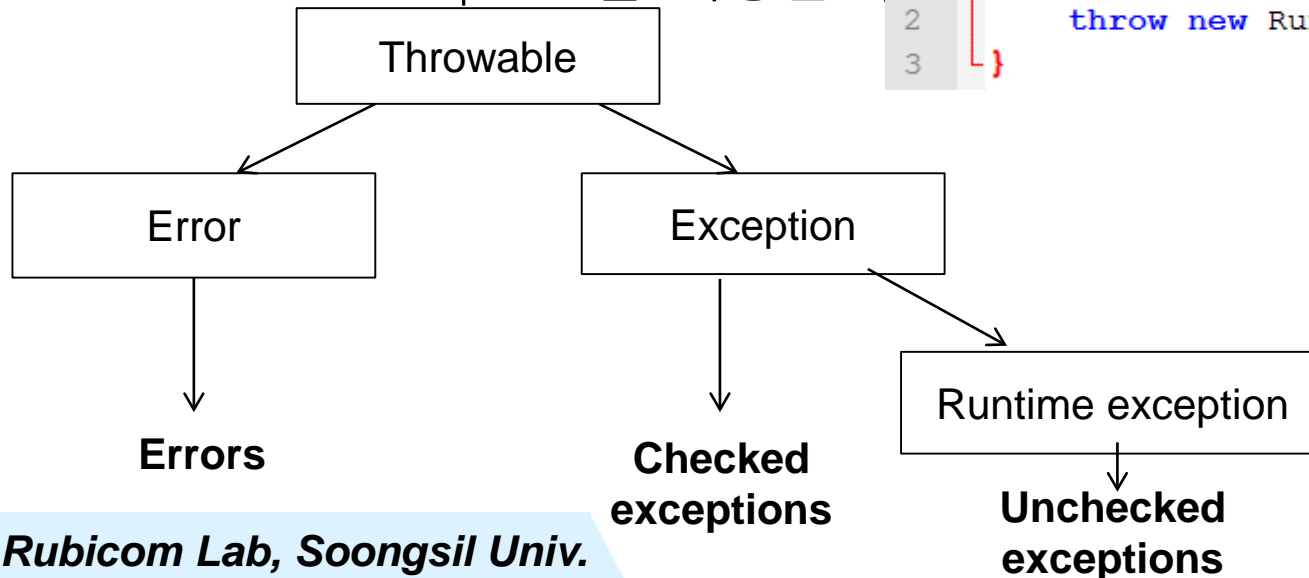
● Exception 관련 클래스 계층구조

- ❖ Error 클래스는 JVM이 전용으로 사용함. 자바 응용이 catch할 수 없음 (예: OutOfMemoryException)
- ❖ 상위 메소드들의 signature에 많은 exception 타입을 열거하는 것 대신에, 해당 응용을 위한 exception tree를 설계하는 것이 필요함

- 예: MyApplicationException이 MyUIException과 MyNetworkException을 포함하도록

- ❖ 이것마저도 불편하면 메소드 signature에 선언할 필요가 없는 runtime exception을 사용할 수

```
1 public void ThrowsRuntimeException() {
2     throw new RuntimeException();
3 }
```



실습: Main2()

```
17 public static void main2(String[] args) { // why no "throws Exception"???
```

```
18     int[][] arrayBlk = {
19         { 0, 1, 0 },
20         { 1, 1, 1 },
21         { 0, 0, 0 }
22     };
23     try {
```

matrix sizes mismatch

ssu.rubicom.tetrismodel.MismatchedMatrixException: matrix sizes mismatch
at first catch

at ssu.rubicom.tetrismodel.Matrix.add([Matrix.java:70](#))

at ssu.rubicom.tetrismodel.Main.main1([Main.java:17](#))

at ssu.rubicom.tetrismodel.Main.main([Main.java:5](#))

```
23     try {
```

```
24         Matrix currBlk = new Matrix(arrayBlk);
```

```
25         Matrix tempBlk = new Matrix(cy:5, cx:5);
```

```
26         //Matrix tempBlk = new Matrix(-1,-1); // falls into the second catch
```

```
27         tempBlk = tempBlk.add(currBlk); // falls into the first catch
```

```
28     } catch(MismatchedMatrixException e) {
```

```
29         e.printStackTrace();
```

```
30         System.out.println(e.getMessage());
```

```
31         System.out.println("at first catch");
```

```
32     } catch(MatrixException e) {
```

```
33         e.printStackTrace();
```

```
34         System.out.println(e.getMessage());
```

```
35         System.out.println("at second catch");
```

```
36     }
```

```
37 }
```

Matrix.java (Exception 적용)

```

3  public class Matrix {
4      private static int nAlloc = 0;
5      private static int nFree = 0;
6      protected void finalize() throws Throwable {...}
10     public int get_nAlloc() { return nAlloc; }
11     public int get_nFree() { return nFree; }
12     private int dy = 0;
13     private int dx = 0;
14     private int[][] array = null;
15     public int get_dy() { return dy; }
16     public int get_dx() { return dx; }
17     public int[][] get_array() { return array; }
18     private void alloc(int cy, int cx) throws MatrixException {
19         if((cy < 0) || (cx < 0))
20             throw new MatrixException("wrong matrix size");
21         dy = cy;
22         dx = cx;
23         array = new int[dy][dx];
24         nAlloc++; // count the number of allocated objects
25     }
26     public Matrix() throws MatrixException { alloc(cy:0, cx:0); }
27     public Matrix(int cy, int cx) throws MatrixException {...}
33     public Matrix(Matrix obj) throws MatrixException {...}
39     public Matrix(int[][] a) throws MatrixException {...}

```


Matrix.java (Exception 적용)

```
45 public Matrix clip(int top, int left, int bottom, int right) throws MatrixException {
46     int cy = bottom - top;
47     int cx = right - left;
48     Matrix temp = new Matrix(cy, cx);
49     for(int y = 0; y < cy; y++){
50         for(int x = 0; x < cx; x++){
51             if((top+y >= 0) && (left+x >= 0) && (top+y < dy) && (left+x < dx))
52                 temp.array[y][x] = array[top+y][left+x];
53             else
54                 throw new MatrixException("invalid matrix range");
55         }
56     }
57     return temp;
58 }
59 public void paste(Matrix obj, int top, int left) throws MatrixException {
60     for(int y = 0; y < obj.dy; y++){
61         for(int x = 0; x < obj.dx; x++) {
62             if((top+y >= 0) && (left+x >= 0) && (top+y < dy) && (left+x < dx))
63                 array[y + top][x + left] = obj.array[y][x];
64             else
65                 throw new MatrixException("invalid matrix range");
66         }
67     }
```

Matrix.java (Exception 적용)

```

68 public Matrix add(Matrix obj) throws MatrixException {
69     if((dx != obj.dx) || (dy != obj.dy))
70         throw new MismatchedMatrixException("matrix sizes mismatch");
71     Matrix temp = new Matrix(dy, dx);
72     for(int y = 0; y < obj.dy; y++)
73         for(int x = 0; x < obj.dx; x++)
74             temp.array[y][x] = array[y][x] + obj.array[y][x];
75     return temp;
76 }
77 public int sum(){...}
84 public void mulc(int coef){...}
89 public boolean anyGreaterThan(int val){...}
97 public void print(){...}
105 // end of Matrix
106 }
107 class MatrixException extends Exception {
108     public MatrixException() { super("Matrix Exception"); }
109     public MatrixException(String msg) { super(msg); }
110 }
111 class MismatchedMatrixException extends MatrixException {
112     public MismatchedMatrixException() { super("Mismatched Matrix Exception"); }
113     public MismatchedMatrixException(String msg) { super(msg); }
114 }

```

Matrix 클래스의 메소드들 이해

● Matrix class의 method들

- ❖ `M.clip(top, left, bottom, right)` : `(top, left, bottom, right)`로 정의되는 사각형에 포함되는 원소들을 또 다른 행렬로 리턴한다.
- ❖ `M.add(obj)`: 행렬 M와 행렬 obj를 원소 대 원소로 더한 결과를 또 다른 행렬로 리턴한다.
- ❖ `M.paste(object, top, left)`: 행렬 M의 `(top, left)` 좌표를 좌상 꼭지점으로 삼아 행렬 object를 행렬 M 안에 복사한다.
- ❖ `M.sum()`: 행렬 M의 모든 원소들의 합을 리턴한다.
- ❖ `M.mulc(coef)`: 행렬 M의 모든 원소에 coef를 곱한다.
- ❖ `M.anyGreaterThan(val)`: 행렬 M 안에 val 값보다 큰 값을 갖는 원소가 하나라도 있으면 true를 리턴한다.

실습: Main3()

```

39 public static void main3(String[] args) throws Exception { // why "throws Exception"???
40     int[][] arrayScreen = {
41         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
42         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
43         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
44         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
45         { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
46         { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },
47     };
48     int[][] arrayBlk = {
49         { 0, 1, 0 },
50         { 1, 1, 1 },
51         { 0, 0, 0 },
52     };
53     Matrix oScreen = new Matrix(arrayScreen);
54     System.out.println("oScreen:");
55     drawMatrix(oScreen); System.out.println();
56
57     Matrix currBlk = new Matrix(arrayBlk);
58     System.out.println("currBlk:");
59     drawMatrix(currBlk); System.out.println();
60
61     int top = 0;
62     int left = 4;
63     Matrix tempBlk = oScreen.clip(top, left, bottom: top+currBlk.get_dy(), right: left+currBlk.get_dx());
64     System.out.println("tempBlk (after clip):");
65     drawMatrix(tempBlk); System.out.println();

```

oScreen:

```

■ □ □ □ □ □ □ □ □ □ □ ■
■ □ □ □ □ □ □ □ □ □ □ ■
■ □ □ □ □ □ □ □ □ □ □ ■
■ □ □ □ □ □ □ □ □ □ □ ■
■ □ □ □ □ □ □ □ □ □ □ ■
■ □ □ □ □ □ □ □ □ □ □ ■
■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■

```

currBlk:

```

□ ■ □
■ ■ ■
□ □ □

```

tempBlk (after clip):

```

□ □ □
□ □ □
□ □ □

```

실습: Main3()

66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84

```
tempBlk = tempBlk.add(currBlk);
System.out.println("tempBlk (after add):");
drawMatrix(tempBlk); System.out.println();

oScreen.paste(tempBlk, top, left);
System.out.println("oScreen (after paste):");
drawMatrix(oScreen); System.out.println();

System.out.println("currBlk.sum()" + currBlk.sum());
System.out.println();

tempBlk.mulc(coef: 2);
System.out.println("tempBlk (after mulc):");
tempBlk.print(); System.out.println();

System.out.println("currBlk.anyGreaterThan(1)=" + currBlk.anyGreaterThan(val: 1));
System.out.println("tempBlk.anyGreaterThan(1)=" + tempBlk.anyGreaterThan(val: 1));
}
```

tempBlk (after add):

```
□ ■ □
■ ■ ■
□ □ □
```

oScreen (after paste):

```
■ □ □ □ □ ■ □ □ □ □ ■
■ □ □ □ ■ ■ ■ □ □ □ ■
■ □ □ □ □ □ □ □ □ □ ■
■ □ □ □ □ □ □ □ □ □ ■
■ □ □ □ □ □ □ □ □ □ ■
■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
```

currBlk.sum()=4

tempBlk (after mulc):

Matrix(3,3)

0 2 0

2 2 2

0 0 0

currBlk.anyGreaterThan(1)=false

tempBlk.anyGreaterThan(1)=true

실습: Main3()

```
85 public static void drawMatrix(Matrix m) {  
86     int dy = m.get_dy();  
87     int dx = m.get_dx();  
88     int array[][] = m.get_array();  
89     for (int y=0; y < dy; y++) {  
90         for (int x=0; x < dx; x++) {  
91             if (array[y][x] == 0) System.out.print("□ ");  
92             else if (array[y][x] == 1) System.out.print("■ ");  
93             else System.out.print("X ");  
94         }  
95         System.out.println();  
96     }  
97 }
```

감사합니다!